

Sintassi Categoriale e Teoria dei Tipi nel Linguaggio Naturale

Introduzione: Fondamenti Matematici della Struttura Linguistica

La sintassi categoriale rappresenta uno dei paradigmi più eleganti e formalmente rigorosi per l'analisi della struttura del linguaggio naturale, radicando la comprensione dei fenomeni sintattici in fondamenta logico-matematiche profonde. Questo approccio, che trova le sue origini nel lavoro pionieristico di Kazimierz Ajdukiewicz negli anni Trenta del Novecento e successivamente nella riflessione di Yehoshua Bar-Hillel, propone una visione della grammatica come sistema di combinazione di categorie sintattiche governato da principi di applicazione funzionale. L'intuizione fondamentale che anima questo programma di ricerca consiste nel riconoscere che le espressioni linguistiche possono essere concepite come entità dotate di un tipo sintattico, e che la combinazione di tali espressioni segue regole che ricordano da vicino la composizione funzionale in matematica e in logica.

La rivoluzione categoriale nella linguistica teorica ha portato con sé una profonda trasformazione nel modo in cui concepiamo la relazione tra forma sintattica e significato semantico. Mentre le grammatiche trasformazionali di stampo chomskyano si concentravano primariamente su derivazioni attraverso regole di riscrittura e trasformazioni strutturali, la grammatica categoriale propone un modello in cui la sintassi e la semantica procedono in parallelo, seguendo un principio di composizionalità rigorosa. Ogni operazione sintattica che combina due espressioni corrisponde a un'operazione semantica che calcola il significato dell'espressione risultante a partire dai significati dei costituenti. Questa visione trova la sua espressione più compiuta nel principio di composizionalità di Frege, secondo il quale il significato di un'espressione complessa è determinato sistematicamente dai significati delle sue parti e dal modo in cui queste sono combinate.

Il passaggio dalla grammatica categoriale classica, con le sue semplici regole di applicazione funzionale, ai sistemi più sofisticati come il calcolo di Lambek e le grammatiche categoriali combinatorie ha rappresentato un momento cruciale nello sviluppo di questa tradizione di ricerca. Joachim Lambek, matematico e logico, propose negli anni Cinquanta un sistema formale che integra la grammatica categoriale con la logica lineare intuizionista non commutativa, creando un quadro in cui le categorie sintattiche sono interpretate come formule di un sistema logico e le derivazioni sintattiche come prove formali. Questo approccio ha rivelato connessioni profonde tra la struttura del linguaggio naturale e sistemi logici fondamentali, aprendo la strada a una comprensione della sintassi come fenomeno computazionale nel senso più astratto del termine.

La teoria dei tipi, sviluppata originariamente da Bertrand Russell come soluzione ai paradossi della teoria degli insiemi e successivamente elaborata da Alonzo Church nel lambda calcolo tipizzato, fornisce il substrato matematico ideale per formalizzare la

semantica composizionale nel contesto categoriale. Nella semantica di Montague, che rappresenta l'applicazione più influente della teoria dei tipi alla linguistica, ogni espressione linguistica è associata a un tipo semantico che specifica la natura della sua denotazione. I tipi base includono tipicamente e per le entità ed t per i valori di verità, mentre i tipi funzionali della forma $(a \rightarrow b)$ denotano funzioni dal tipo a al tipo b . Questa architettura permette di rappresentare in modo preciso la semantica di categorie sintattiche complesse: i verbi transitivi, per esempio, hanno tipo $(e \rightarrow (e \rightarrow t))$, indicando che prendono due argomenti di tipo e e restituiscono un valore di verità.

Il Calcolo di Lambek: Fondamenti Logici della Sintassi

Il calcolo sintattico di Lambek, spesso indicato con la sigla L , costituisce un sistema formale che integra la grammatica categoriale con la logica lineare intuizionista. La caratteristica distintiva di questo calcolo risiede nella sua interpretazione delle categorie sintattiche come formule di un linguaggio logico e delle derivazioni sintattiche come dimostrazioni formali. Lambek propose il suo sistema nel 1958 nell'articolo "The Mathematics of Sentence Structure", gettando le basi per una tradizione di ricerca che avrebbe rivelato connessioni profonde tra sintassi linguistica, teoria della dimostrazione e teoria delle categorie.

Nel calcolo di Lambek, le categorie sintattiche sono costruite a partire da un insieme finito di categorie primitive mediante l'uso di due operatori di formazione di categorie: lo slash sinistro $()$ e lo slash destro $()$. Se A e B sono categorie sintattiche, allora A/B rappresenta la categoria di un'espressione che cerca un argomento di categoria B alla sua destra per formare un'espressione di categoria A , mentre $B \backslash A$ rappresenta la categoria di un'espressione che cerca un argomento di categoria B alla sua sinistra per formare un'espressione di categoria A . Questa notazione cattura elegantemente l'intuizione che molte espressioni linguistiche sono incomplete e richiedono argomenti per formare costituenti completi.

Consideriamo esempi concreti per illustrare questa architettura. Un sintagma nominale come "Maria" o "il gatto" è tipicamente assegnato alla categoria primitiva NP . Un verbo intransitivo come "dorme" cerca un soggetto alla sua sinistra per formare una frase completa, dunque riceve la categoria $NP \backslash S$, dove S indica la categoria delle frasi. Un verbo transitivo come "ama" richiede prima un oggetto alla sua destra e poi un soggetto alla sua sinistra, ricevendo la categoria $(NP \backslash S) / NP$. Questa rappresentazione cattura la valenza del verbo e l'ordine in cui i suoi argomenti devono essere saturati.

Le regole di inferenza del calcolo di Lambek specificano come le categorie possono essere combinate per formare derivazioni sintattiche valide. Il sistema include regole di eliminazione per gli slash che corrispondono all'applicazione funzionale, nonché regole di introduzione che permettono di astrarre su posizioni di argomento. La regola di eliminazione dello slash destro, indicata come $/E$, stabilisce che se abbiamo un'espressione di categoria A/B seguita da un'espressione di categoria B , possiamo derivare un'espressione di categoria A . Formalmente, questo si scrive come:

$$\frac{X \vdash A/B \quad Y \vdash B}{X \cdot Y \vdash A} (/E)$$

dove X e Y rappresentano sequenze di espressioni e $\cdot$ denota la concatenazione. Analogamente, la regola di eliminazione dello slash sinistro $\backslash E$ permette di combinare un'espressione di categoria B con una successiva espressione di categoria $B \backslash A$ per ottenere A :

$$\frac{X \vdash B \quad Y \vdash B \backslash A}{X \cdot Y \vdash A} (\backslash E)$$

Queste regole di eliminazione catturano l'intuizione di base dell'applicazione funzionale nella grammatica categoriale: un funtore si combina con il suo argomento per produrre il risultato. Tuttavia, il calcolo di Lambek va oltre la semplice applicazione includendo anche regole di introduzione che permettono di costruire funtori attraverso un processo di astrazione. La regola di introduzione dello slash destro $/I$ permette di derivare A/B da una derivazione ipotetica in cui assumiamo la disponibilità di B per derivare A :

$$\frac{X \cdot [B] \vdash A}{X \vdash A/B} (/I)$$

dove $[B]$ indica un'ipotesi scaricata. Questa regola corrisponde all'astrazione lambda nel lambda calcolo e permette di catturare fenomeni di dipendenza sintattica a lunga distanza.

Un aspetto fondamentale del calcolo di Lambek è la sua natura non commutativa: l'ordine delle espressioni nella sequenza è significativo e non può essere permutato arbitrariamente. Questo riflette il fatto che il linguaggio naturale è sensibile all'ordine delle parole, e che la sequenza "Maria ama Giovanni" ha una struttura sintattica diversa da "Giovanni ama Maria", anche se coinvolge le stesse espressioni lessicali. La non commutatività è codificata nel fatto che le regole di inferenza preservano l'ordine delle espressioni nelle sequenze.

Il calcolo di Lambek può essere arricchito con un operatore di prodotto $(\cdot)$ che permette di formare categorie congiunte, ottenendo quello che viene chiamato il calcolo di prodotto di Lambek o LP. Il prodotto categoriale $A \cdot B$ rappresenta la categoria di un'espressione che consiste nella giustapposizione di un'espressione di categoria A seguita da un'espressione di categoria B . Questo operatore è utile per modellare fenomeni come i verbi frasali in inglese o le costruzioni idiomatiche che comportano sequenze fisse di elementi.

La semantica del calcolo di Lambek può essere fornita attraverso varie interpretazioni. Un approccio standard consiste nell'interpretare le categorie come tipi nel lambda calcolo tipizzato e le derivazioni come lambda termini. In questa interpretazione, lo slash destro A/B corrisponde al tipo funzionale $B \rightarrow A$, mentre lo slash sinistro $B \backslash A$ richiede un trattamento più sottile che può coinvolgere l'inversione dell'ordine degli argomenti o l'uso di combinatori appropriati. La regola di eliminazione dello slash corrisponde all'applicazione di funzione, mentre la regola di introduzione corrisponde all'astrazione lambda.

Un'altra interpretazione semantica importante è quella categoriale, in cui le categorie sintattiche sono interpretate come oggetti in una categoria appropriata e i funtori sintattici come morfismi. Questa prospettiva rivela che il calcolo di Lambek è intimamente connesso con la struttura delle categorie chiuse rispetto a un monoide, fornendo una caratterizzazione algebrica astratta della sintassi. In particolare, il calcolo di Lambek è sound e complete rispetto a modelli in categorie chiuse monoidali, un risultato che stabilisce una corrispondenza profonda tra derivabilità sintattica e validità semantica.

La teoria della dimostrazione del calcolo di Lambek è stata studiata estensivamente, rivelando proprietà formali importanti. Il sistema gode della proprietà di eliminazione del taglio, che garantisce che ogni derivazione può essere trasformata in una forma normale in cui non appaiono "lemmi" intermedi. Questa proprietà è cruciale per applicazioni computazionali, poiché implica che esiste un algoritmo di decisione efficiente per determinare se una data sequenza di categorie è derivabile. Il teorema di eliminazione del taglio per il calcolo di Lambek è stato dimostrato da Lambek stesso e rappresenta uno dei risultati fondamentali della logica categoriale.

Dal punto di vista della complessità computazionale, il problema di determinare se una sequenza di categorie è derivabile nel calcolo di Lambek è NP-completo, un risultato dimostrato da Pentus negli anni Novanta. Questo significa che, nel caso peggiore, il tempo richiesto per decidere la derivabilità può crescere esponenzialmente con la dimensione dell'input. Tuttavia, per grammatiche specifiche e con opportune restrizioni, è possibile ottenere algoritmi di parsing più efficienti, e infatti sono stati sviluppati parser basati su chart e su tecniche di programmazione dinamica che funzionano ragionevolmente bene in pratica.

Il calcolo di Lambek è stato esteso in varie direzioni per catturare fenomeni linguistici più complessi. Il calcolo di Lambek-Grishin, proposto da Grishin e sviluppato ulteriormente da vari ricercatori, aggiunge operatori duali ai connettivi di Lambek, permettendo di modellare fenomeni di dipendenza multipla e estrazione. Altre estensioni includono l'aggiunta di modalità per controllare le proprietà strutturali della logica sottostante, come la commutatività e l'associatività, ottenendo i cosiddetti calcoli multimodali che possono catturare variazioni cross-linguistiche nell'ordine delle parole e nella struttura sintattica.

Grammatiche Categoriali Combinatorie: Flessibilità Sintattica e Costituenza

Le grammatiche categoriali combinatorie, sviluppate principalmente da Mark Steedman a partire dagli anni Ottanta, rappresentano una variante della grammatica categoriale che enfatizza la flessibilità nella derivazione sintattica e la stretta corrispondenza tra struttura sintattica superficiale e struttura informazionale. A differenza del calcolo di Lambek, che si basa su un insieme ristretto di regole di inferenza logica, le CCG (Combinatory Categorical Grammar) incorporano un repertorio più ampio di regole combinatorie che permettono costituenti "non standard" e derivazioni multiple per la stessa stringa, riflettendo l'idea che la struttura sintattica sia determinata in parte da fattori pragmatici e prosodici.

Il punto di partenza delle CCG è simile a quello della grammatica categoriale classica: ogni espressione lessicale è associata a una categoria sintattica che specifica il tipo di

costituente che forma e gli argomenti che richiede. Le categorie sono costruite da categorie atomiche come S (frase), N (nome) e NP (sintagma nominale) mediante operatori slash. Tuttavia, le CCG arricchiscono questo sistema con un insieme di regole combinatorie che vanno oltre la semplice applicazione funzionale.

La regola di applicazione funzionale in avanti (forward application, indicata con $>$) permette a un funtore che cerca un argomento alla sua destra di combinarsi con quell'argomento:

$$\begin{array}{l} X: A/B \quad Y: B \\ \hline X Y: A \end{array} (>)$$

Analogamente, l'applicazione funzionale all'indietro (backward application, $<$) gestisce funtori che cercano argomenti alla loro sinistra:

$$\begin{array}{l} X: B \quad Y: B \backslash A \\ \hline X Y: A \end{array} (<)$$

Queste regole corrispondono alle eliminazioni degli slash nel calcolo di Lambek. Tuttavia, le CCG introducono anche regole di composizione funzionale che permettono di combinare due funtori per creare un funtore più complesso. La composizione in avanti (forward composition, indicata con $>B$) combina un funtore A/B con un funtore B/C per creare un funtore A/C :

$$\begin{array}{l} X: A/B \quad Y: B/C \\ \hline X Y: A/C \end{array} (>B)$$

Semanticamente, questa regola corrisponde alla composizione di funzioni: se f ha tipo $(B \rightarrow A)$ e g ha tipo $(C \rightarrow B)$, allora la loro composizione $f \circ g$ ha tipo $(C \rightarrow A)$. La composizione all'indietro ($<B$) opera in modo analogo ma con funtori che cercano argomenti alla sinistra.

Un'altra regola cruciale è la type-raising (innalzamento di tipo), che permette di convertire un argomento in un funtore che cerca un funtore che richiede quell'argomento. Per esempio, un sintagma nominale NP può essere innalzato a $S/(S \backslash NP)$, che cerca un predicato verbale per formare una frase. Questa regola è particolarmente importante per modellare fenomeni di coordinazione e di dipendenza a lunga distanza. Il type-raising cattura l'intuizione che un argomento può comportarsi come se fosse esso stesso un funtore in certi contesti sintattici.

La flessibilità delle CCG si manifesta nel fatto che una stessa stringa può ricevere multiple derivazioni strutturali, tutte sintatticamente legittime, ma che differiscono nella loro costituenza superficiale. Questa proprietà viene chiamata "equivalenza derivazionale spuria" (spurious ambiguity) ed è caratteristica distintiva delle CCG. Steedman argomenta che questa apparente ambiguità è in realtà un pregio, non un difetto, del formalismo, poiché le

diverse derivazioni corrispondono a diverse strutture informative e prosodiche della stessa frase.

Consideriamo l'esempio dell'analisi di una frase come "Maria pensa che Giovanni dormirà". In una derivazione standard, il verbo "pensa" si combina prima con la frase subordinata "che Giovanni dormirà" per formare il predicato verbale "pensa che Giovanni dormirà", che poi si combina con il soggetto "Maria". Tuttavia, utilizzando il type-raising e la composizione, è possibile derivare una struttura in cui "Maria pensa" forma prima un costituente, che poi si combina con "che Giovanni dormirà". Entrambe le derivazioni sono valide e producono la stessa interpretazione semantica, ma riflettono diverse strutture informative: la prima è neutra, mentre la seconda potrebbe corrispondere a una lettura in cui "Maria pensa" è in focus contrastivo.

Le CCG hanno dimostrato di essere particolarmente efficaci nel trattamento di fenomeni sintattici complessi che hanno posto sfide alle teorie sintattiche tradizionali. L'estrazione a lunga distanza, per esempio, può essere modellata elegantemente senza ricorrere a tracce o movimenti. Nella frase "Quale libro pensi che Maria abbia letto?", il sintagma interrogativo "quale libro" può essere analizzato come avente una categoria che cerca una frase con un "buco" nella posizione dell'oggetto. Attraverso l'uso della composizione funzionale, questo buco può essere propagato attraverso vari livelli di embedding sintattico fino a essere saturato nella posizione appropriata.

La coordinazione è un altro fenomeno che le CCG trattano in modo particolarmente naturale. La regola di coordinazione nelle CCG permette di coordinare costituenti della stessa categoria sintattica, anche se questi costituenti non sarebbero tradizionalmente considerati come tali nelle grammatiche a struttura sintagmatica. Per esempio, nella frase "Maria cucinò e Giovanni mangiò la cena", i costituenti "Maria cucinò" e "Giovanni mangiò" possono essere coordinati se vengono entrambi analizzati come funtori che cercano un oggetto alla destra, ottenibile attraverso type-raising e composizione. Questo permette di derivare costruzioni di coordinazione "non costituente" senza postulare ellissi o altre operazioni trasformazionali.

Dal punto di vista della semantica, le CCG implementano il principio di trasparenza semantica in modo rigoroso: ogni regola combinatoria sintattica è associata a una regola di composizione semantica che specifica come calcolare il significato dell'espressione risultante. L'applicazione funzionale corrisponde all'applicazione di funzione nella semantica, la composizione corrisponde alla composizione di funzioni, e il type-raising corrisponde a una trasformazione del significato che preserva l'interpretazione fattuale ma cambia il tipo semantico. Questa architettura garantisce che la sintassi e la semantica procedano in perfetto parallelismo.

Le CCG sono state implementate in vari sistemi di parsing computazionale e hanno dimostrato di essere competitive con altri formalismi in termini di copertura empirica e efficienza computazionale. Il CCGbank, un corpus annotato derivato dal Penn Treebank con categorie CCG, ha fornito una risorsa preziosa per lo sviluppo e la valutazione di parser statistici basati su CCG. Questi parser utilizzano tecniche di apprendimento automatico per stimare le probabilità delle varie categorie lessicali e delle regole combinatorie, permettendo di disambiguare tra le multiple derivazioni possibili sulla base di dati empirici.

Un aspetto importante delle CCG è il loro approccio lessicalista: gran parte dell'informazione sintattica è codificata nel lessico attraverso l'assegnazione di categorie complesse alle espressioni lessicali, piuttosto che in regole sintattiche astratte. Questo rende le CCG particolarmente adatte per modellare la variazione cross-linguistica, poiché le differenze tra lingue possono essere catturate in gran parte attraverso differenze nell'inventario di categorie lessicali. Per esempio, le differenze nell'ordine delle parole tra inglese e giapponese possono essere modellate assegnando categorie con slash orientati diversamente: mentre i verbi inglesi cercano oggetti alla destra, i verbi giapponesi cercano oggetti alla sinistra.

Le CCG hanno anche trovato applicazioni nella modellazione della prosodia e dell'interfaccia sintassi-fonologia. Steedman ha proposto che le diverse derivazioni CCG di una stessa stringa corrispondano a diverse strutture prosodiche, con i costituenti sintattici che determinano i domini per le operazioni fonologiche come l'assegnazione dell'accento. Questa prospettiva fornisce un quadro unificato per comprendere come la struttura sintattica, informazionale e prosodica interagiscono nella produzione e nella comprensione del linguaggio.

Teoria dei Tipi e Lambda Calcolo Tipizzato nella Semantica

La teoria dei tipi costituisce il fondamento matematico per una semantica composizionale rigorosa del linguaggio naturale, fornendo un sistema formale che permette di classificare le espressioni in base alla natura delle loro denotazioni e di specificare regole precise per la composizione del significato. Il lambda calcolo tipizzato, sviluppato da Alonzo Church negli anni Trenta e quaranta come formalizzazione della nozione di funzione computabile, è stato adattato da Richard Montague negli anni Settanta per creare un quadro semantico che potesse competere in rigore e precisione con le teorie sintattiche formali dell'epoca.

Nella semantica montaguiana, ogni espressione linguistica è associata a un tipo semantico che specifica la categoria ontologica della sua denotazione. I tipi base sono tipicamente e per le entità (individui nel dominio del discorso) ed t per i valori di verità (vero e falso). A partire da questi tipi base, si costruiscono tipi complessi utilizzando un'operazione di formazione di tipi funzionali: se a e b sono tipi, allora $(a \rightarrow b)$ è il tipo delle funzioni che mappano elementi di tipo a in elementi di tipo b . Questa architettura permette di costruire una gerarchia di tipi che riflette la struttura compositiva del significato.

I sostantivi comuni sono tipicamente interpretati come funzioni da entità a valori di verità, quindi hanno tipo $(e \rightarrow t)$. Questa scelta riflette l'idea che un sostantivo come "gatto" denota la proprietà di essere un gatto, che può essere concepita come una funzione che restituisce vero per ogni entità che è un gatto e falso altrimenti. I nomi propri, d'altra parte, denotano direttamente individui e quindi hanno tipo e . I verbi intransitivi, come i sostantivi comuni, denotano proprietà e hanno tipo $(e \rightarrow t)$, mentre i verbi transitivi denotano relazioni binarie tra individui e quindi hanno tipo $(e \rightarrow (e \rightarrow t))$, che può essere pensato come una funzione che prende un individuo (l'oggetto) e restituisce una proprietà (la proprietà di stare nella relazione con quell'oggetto).

I determinanti come "ogni", "qualche" e "il" giocano un ruolo cruciale nella semantica montaguiana e ricevono tipi che riflettono la loro funzione di combinare sostantivi con predicati verbali per formare frasi. Nella teoria dei quantificatori generalizzati, sviluppata da Montague e successivamente elaborata da Barwise, Cooper e altri, i determinanti sono interpretati come funzioni di ordine superiore che prendono due proprietà (quella espressa dal sostantivo e quella espressa dal predicato verbale) e restituiscono un valore di verità. Il tipo di un determinante è quindi $((e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t))$, abbreviato spesso come $(1 \rightarrow (1 \rightarrow t))$ dove 1 è una notazione alternativa per $(e \rightarrow t)$.

Il lambda calcolo fornisce la sintassi per rappresentare funzioni e per specificare come queste si applicano ai loro argomenti. Un'astrazione lambda $\lambda x.\phi$ denota una funzione che, quando applicata a un argomento a , restituisce il risultato di sostituire a al posto di x in ϕ . Per esempio, l'espressione lambda $\lambda x.\text{gatto}(x)$ denota la proprietà di essere un gatto, mentre $\lambda x.\lambda y.\text{ama}(y,x)$ denota la relazione di amare, concepita come una funzione che prende prima l'oggetto dell'amore e poi il soggetto.

La regola fondamentale di valutazione nel lambda calcolo è la beta-riduzione, che specifica come valutare l'applicazione di una funzione a un argomento:

$$(\lambda x.\phi)(a) \rightarrow \phi[x := a]$$

dove $\phi[x := a]$ indica il risultato di sostituire tutte le occorrenze libere di x in ϕ con a , con opportune precauzioni per evitare catture di variabili. Questa regola permette di calcolare il significato di espressioni complesse riducendole a forme normali attraverso applicazioni successive della beta-riduzione.

Consideriamo un esempio concreto di derivazione semantica. La frase "Ogni gatto dorme" può essere analizzata assegnando i seguenti significati alle sue parti:

- "gatto": $\lambda x.\text{gatto}(x)$ di tipo $(e \rightarrow t)$
- "dorme": $\lambda x.\text{dorme}(x)$ di tipo $(e \rightarrow t)$
- "ogni": $\lambda P.\lambda Q.\forall x[P(x) \rightarrow Q(x)]$ di tipo $((e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t))$

La derivazione procede componendo questi significati secondo la struttura sintattica. Il determinante "ogni" si combina prima con il sostantivo "gatto" attraverso applicazione funzionale:

$$(\lambda P.\lambda Q.\forall x[P(x) \rightarrow Q(x)])(\lambda x.\text{gatto}(x))$$

Applicando la beta-riduzione:

$$\lambda Q.\forall x[\text{gatto}(x) \rightarrow Q(x)]$$

Questo rappresenta il significato del sintagma nominale quantificato "ogni gatto", che è una funzione che prende una proprietà Q e restituisce vero se e solo se ogni gatto ha la proprietà Q . Questo sintagma si combina poi con il predicato verbale "dorme":

$$(\lambda Q. \forall x[\text{gatto}(x) \rightarrow Q(x)])(\lambda y.\text{dorme}(y))$$

Applicando nuovamente la beta-riduzione:

$$\forall x[\text{gatto}(x) \rightarrow \text{dorme}(x)]$$

che è esattamente la formula logica che esprime che ogni gatto dorme.

Il lambda calcolo tipizzato impone restrizioni sulla formazione di espressioni per garantire che solo combinazioni semanticamente sensate siano permesse. Un'applicazione $(f\ a)$ è ben tipizzata solo se f ha un tipo funzionale $(b \rightarrow c)$ e a ha tipo b , e in tal caso l'applicazione ha tipo c . Un'astrazione $\lambda x.\varphi$ è ben tipizzata solo se φ è ben tipizzata assumendo che x abbia un certo tipo a , e in tal caso l'astrazione ha tipo $(a \rightarrow b)$ dove b è il tipo di φ . Queste regole di tipizzazione prevengono paradossi e garantiscono che ogni espressione ben formata denoti un oggetto matematico ben definito.

La teoria dei tipi può essere arricchita in varie direzioni per catturare fenomeni semantici più complessi. Il sistema di tipi di Montague è di ordine superiore, permettendo quantificazione su proprietà e relazioni oltre che su individui. Questo è essenziale per modellare fenomeni come gli avverbi di quantificazione e i verbi di attitudine proposizionale. Per esempio, un avverbio come "sempre" può essere interpretato come un quantificatore su eventi, richiedendo un tipo che permette di quantificare su proprietà di eventi.

Un'estensione importante è l'aggiunta di tipi per entità intensionali, necessari per trattare contesti opachi e atteggiamenti proposizionali. Nella semantica intensionale, sviluppata da Montague e successivamente raffinata da molti altri, le espressioni non denotano direttamente estensioni ma intensioni, concepite come funzioni da mondi possibili (e istanti temporali) a estensioni. Un tipo intensionale $\langle a \rangle$ denota funzioni da situazioni a denotazioni di tipo a . Questo framework permette di modellare la distinzione tra riferimento *de dicto* e *de re*, nonché fenomeni come la sostitutività in contesti intensionali.

Il polimorfismo è un altro arricchimento importante della teoria dei tipi che trova applicazioni nella semantica del linguaggio naturale. In un sistema polimorfico, alcune espressioni possono avere tipi variabili che vengono istanziati in modi diversi in contesti diversi. Questo è utile per modellare fenomeni come la coordinazione polimorfica, dove costituenti di categorie sintattiche diverse possono essere coordinati se condividono una struttura tipo sottostante.

La corrispondenza di Curry-Howard, scoperta indipendentemente da Haskell Curry e William Howard, stabilisce un isomorfismo profondo tra sistemi di tipi e sistemi logici: i tipi corrispondono a formule logiche, i termini a dimostrazioni, e la beta-riduzione a eliminazione del taglio. Questa corrispondenza fornisce una prospettiva unificata su computazione e deduzione, rivelando che programmare è provare e che provare è programmare. Nel contesto della linguistica, la corrispondenza di Curry-Howard suggerisce che le derivazioni sintattiche nel calcolo di Lambek corrispondono a termini lambda nel calcolo tipizzato, e che la composizione semantica può essere vista come valutazione di programmi.

La teoria dei tipi dipendenti, in cui i tipi possono dipendere da valori, ha recentemente attirato l'attenzione nella semantica formale come strumento per modellare fenomeni di anafora e dipendenza contestuale. In un sistema con tipi dipendenti, possiamo avere tipi come "il tipo dei gatti di x " che dipende da un particolare individuo x , permettendo di esprimere dipendenze semantiche sottili che sono difficili da catturare nei sistemi di tipi semplici.

Teoria delle Categorie e Composizione Sintattica

La teoria delle categorie, sviluppata da Samuel Eilenberg e Saunders Mac Lane negli anni Quaranta come linguaggio per esprimere analogie strutturali in diverse aree della matematica, fornisce un quadro unificante per comprendere la composizione sintattica e semantica nel linguaggio naturale. L'intuizione fondamentale della teoria delle categorie è che ciò che conta non sono tanto gli oggetti individuali quanto le relazioni (morfismi) tra di essi e come queste relazioni si compongono. Questo focus sulla struttura compositiva si allinea perfettamente con il principio di composizionalità che sta al cuore della semantica formale.

Una categoria C consiste di una collezione di oggetti $\text{Ob}(C)$, una collezione di morfismi (o frecce) tra oggetti, e un'operazione di composizione dei morfismi che soddisfa determinate proprietà. Per ogni coppia di oggetti A e B , esiste un insieme $C(A,B)$ di morfismi da A a B .

Per ogni tripla di oggetti A , B e C , deve esistere un'operazione di composizione che prende un morfismo $f: A \rightarrow B$ e un morfismo $g: B \rightarrow C$ e produce un morfismo $g \circ f: A \rightarrow C$. Questa composizione deve essere associativa, nel senso che per morfismi $f: A \rightarrow B$, $g: B \rightarrow C$ e $h: C \rightarrow D$, deve valere $(h \circ g) \circ f = h \circ (g \circ f)$. Inoltre, per ogni oggetto A deve esistere un morfismo identità $\text{id}_A: A \rightarrow A$ tale che per ogni morfismo $f: A \rightarrow B$ vale $f \circ \text{id}_A = f$ e $\text{id}_B \circ f = f$.

Nel contesto della semantica del linguaggio naturale, possiamo costruire la categoria Sem in cui gli oggetti sono tipi semantici e i morfismi sono funzioni semantiche. Per esempio, esiste un morfismo dal tipo e al tipo t per ogni proprietà nel dominio, e questi morfismi si compongono nel modo ovvio: se f è una funzione da e a t e g è una funzione da t a t , allora $g \circ f$ è una funzione da e a t . Questa prospettiva categoriale rivela che la composizione semantica è semplicemente composizione di morfismi nella categoria appropriata.

Le categorie chiuse monoidali giocano un ruolo particolarmente importante nella semantica categoriale del linguaggio naturale. Una categoria monoidale è dotata di un'operazione di prodotto tensoriale $\otimes$ che combina oggetti, insieme a un oggetto unità I , soddisfacendo appropriate leggi di associatività e unità a meno di isomorfismi canonici. Una categoria è chiusa se per ogni oggetto B esiste un funtore di esponenziazione $[B, -]$ che è aggiunto a destra al funtore $- \otimes B$. Questo significa che esiste una biiezione naturale tra morfismi $A \otimes B \rightarrow C$ e morfismi $A \rightarrow [B, C]$.

La rilevanza delle categorie chiuse monoidali per la linguistica deriva dal fatto che il calcolo di Lambek può essere interpretato come la logica interna di tali categorie. In questa interpretazione, le categorie sintattiche corrispondono a oggetti nella categoria, le derivazioni

sintattiche a morfismi, il prodotto categoriale $\cdot$ corrisponde al prodotto tensoriale $\otimes$, e gli slash $/$ e $\backslash$ corrispondono agli esponenziali destro e sinistro che catturano la struttura di chiusura. La regola di eliminazione dello slash corrisponde all'applicazione dell'aggiunzione, mentre la regola di introduzione corrisponde alla trasposta dell'aggiunzione.

Più precisamente, il calcolo di Lambek è sound e complete rispetto a modelli in categorie chiuse monoidali residue, dove la residuazione cattura la relazione tra il prodotto e gli esponenziali. Un residuo destro di $\otimes$ rispetto a un oggetto B è un funtore $[B, -]$ tale che esiste una biiezione naturale tra $A \otimes B \sqsubseteq C$ e $A \sqsubseteq [B, C]$, dove $\sqsubseteq$ indica l'ordine nella categoria (vista come categoria preordinata). Analogamente, un residuo sinistro è un funtore $[-, B]$ tale che $B \otimes A \sqsubseteq C$ se e solo se $A \sqsubseteq [-, B]C$. Questi residui corrispondono esattamente agli slash destro e sinistro del calcolo di Lambek.

Un esempio concreto di categoria chiusa monoidale rilevante per la semantica è la categoria $\mathbf{Rel}$ di insiemi e relazioni. Gli oggetti sono insiemi, i morfismi da A a B sono relazioni $R \subseteq A \times B$, la composizione è la composizione di relazioni, il prodotto tensoriale è il prodotto cartesiano di insiemi, e gli esponenziali sono costruiti usando quozienti appropriati. Questa categoria fornisce modelli per interpretazioni relazionali del significato, che sono utili per modellare fenomeni come la vaghezza e la sottospecificazione.

I funtori tra categorie forniscono un modo per confrontare e collegare diverse strutture categoriali. Un funtore $F: C \rightarrow D$ mappa oggetti di C in oggetti di D e morfismi di C in morfismi di D preservando l'identità e la composizione: $F(\text{id}_A) = \text{id}_{F(A)}$ e $F(g \circ f) = F(g) \circ F(f)$. Nel contesto linguistico, i funtori possono rappresentare traduzioni tra livelli di rappresentazione o mappature tra lingue diverse. Per esempio, un funtore dalla categoria delle derivazioni sintattiche alla categoria dei significati semantici codifica l'interpretazione semantica compositiva.

Le trasformazioni naturali tra funtori forniscono un livello ancora più elevato di astrazione. Una trasformazione naturale $\alpha: F \Rightarrow G$ tra funtori $F, G: C \rightarrow D$ consiste di una famiglia di morfismi $\alpha_A: F(A) \rightarrow G(A)$ per ogni oggetto A in C , tale che per ogni morfismo $f: A \rightarrow B$ in C il diagramma commuta: $G(f) \circ \alpha_A = \alpha_B \circ F(f)$. Le trasformazioni naturali possono modellare sistematiche relazioni tra diverse interpretazioni semantiche o tra diversi livelli di rappresentazione linguistica.

La teoria delle categorie ha anche influenzato lo sviluppo di grammatiche categoriali astratte (abstract categorial grammars), introdotte da Philippe de Groote. In questo framework, una grammatica è specificata attraverso due componenti: una firma astratta che definisce un vocabolario di costanti tipizzate nel lambda calcolo di ordine superiore, e un lessico che mappa queste costanti astratte in forme concrete (sintattiche, fonologiche, ecc.) attraverso omomorfismi. Questo approccio permette di separare chiaramente la struttura logica del significato dalla sua realizzazione superficiale, facilitando analisi multilingue e studi tipologici.

Un concetto importante nella teoria categoriale della sintassi è quello di grammatica pregroup, introdotta da Joachim Lambek come semplificazione del suo calcolo originale. Una grammatica pregroup è basata su un pregroup libero, una struttura algebrica che generalizza i gruppi rilassando gli assiomi di annullamento. In un pregroup, ogni elemento a ha un aggiunto destro a^r e un aggiunto sinistro a^l tali che $a \cdot a^r \sqsubseteq 1 \sqsubseteq a^r \cdot a$ e $a^l \cdot a \sqsubseteq 1 \sqsubseteq a \cdot a^l$.

$a \cdot a^\dagger$, dove $\sqsubseteq$ è un preordine parziale e 1 è l'elemento unità. Le derivazioni in una grammatica pregroup sono sequenze di riduzioni che trasformano una stringa di tipi nel tipo obiettivo (tipicamente s per frase) usando le proprietà di aggiunzione.

Le grammatiche pregroup hanno il vantaggio di essere computazionalmente più trattabili rispetto al calcolo di Lambek completo, pur mantenendo sufficiente espressività per catturare molti fenomeni sintattici interessanti. La decidibilità del parsing è polinomiale per le grammatiche pregroup, a differenza della NP-completezza del calcolo di Lambek generale. Questo le rende attraenti per applicazioni pratiche di elaborazione del linguaggio naturale.

La semantica categoriale ha anche ispirato approcci diagrammatici alla composizione del significato, in cui le derivazioni sintattiche sono rappresentate come diagrammi di stringa (string diagrams) in categorie monoidali. In questa rappresentazione, gli oggetti sono rappresentati come fili e i morfismi come nodi che connettono fili, con la composizione sequenziale rappresentata dalla connessione verticale di diagrammi e il prodotto tensoriale dalla giustapposizione orizzontale. Questi diagrammi forniscono un'interfaccia visuale intuitiva per ragionare sulla composizione semantica e hanno trovato applicazioni nel calcolo quantistico e nella semantica distribuzionale.

Corrispondenza di Curry-Howard tra Logica e Computazione Linguistica

La corrispondenza di Curry-Howard, nota anche come isomorfismo di Curry-Howard o paradigma proposizioni-come-tipi, stabilisce una connessione profonda tra sistemi logici e sistemi di programmazione tipizzati. Questa corrispondenza rivela che le formule logiche corrispondono a tipi, le dimostrazioni logiche corrispondono a programmi, e la semplificazione di dimostrazioni corrisponde all'esecuzione di programmi. Nel contesto della linguistica computazionale, questa corrispondenza fornisce un ponte concettuale tra la derivazione sintattica, concepita come dimostrazione in un sistema logico come il calcolo di Lambek, e la computazione semantica, concepita come valutazione di un programma nel lambda calcolo tipizzato.

La forma base della corrispondenza di Curry-Howard mette in relazione la logica proposizionale intuizionista con il lambda calcolo semplicemente tipizzato. In questa corrispondenza, una formula $A \rightarrow B$ corrisponde al tipo funzionale $A \rightarrow B$, e una dimostrazione di $A \rightarrow B$ corrisponde a una funzione che trasforma prove di A in prove di B . La regola di introduzione dell'implicazione nella logica naturale deduttiva corrisponde all'astrazione lambda, mentre la regola di eliminazione dell'implicazione (modus ponens) corrisponde all'applicazione di funzione.

Più precisamente, consideriamo il sistema di deduzione naturale per la logica proposizionale intuizionista. La regola di introduzione dell'implicazione afferma che se possiamo derivare B assumendo A , allora possiamo derivare $A \rightarrow B$ scaricando l'assunzione:

[A]
 $\vdots$
 B

----- ($\rightarrow I$)
 $A \rightarrow B$

Questa regola corrisponde esattamente alla formazione di un'astrazione lambda: se abbiamo un termine t di tipo B che può contenere una variabile libera x di tipo A , possiamo formare il termine $\lambda x.t$ di tipo $A \rightarrow B$. La regola di eliminazione dell'implicazione:

$A \rightarrow B \quad A$
 ----- ($\rightarrow E$)
 B

corrisponde all'applicazione di funzione: se abbiamo un termine f di tipo $A \rightarrow B$ e un termine a di tipo A , possiamo formare il termine $f a$ di tipo B .

Le regole di introduzione ed eliminazione della congiunzione logica corrispondono a operazioni su prodotti di tipi. Se $A \wedge B$ corrisponde al tipo prodotto $A \times B$, allora una dimostrazione di $A \wedge B$ corrisponde a una coppia (a,b) dove a è una prova di A e b è una prova di B . Le regole di eliminazione della congiunzione, che permettono di estrarre A da $A \wedge B$ e B da $A \wedge B$, corrispondono alle proiezioni π_1 e π_2 che estraggono i componenti di una coppia.

La disgiunzione logica $A \vee B$ corrisponde al tipo somma $A + B$, e una dimostrazione di $A \vee B$ è o una prova di A etichettata come proveniente dalla sinistra, o una prova di B etichettata come proveniente dalla destra. L'eliminazione della disgiunzione, che permette di derivare C da $A \vee B$ fornendo derivazioni da A a C e da B a C , corrisponde a un costrutto case che analizza una somma e applica la funzione appropriata a seconda del caso.

La falsità logica $\perp$ corrisponde al tipo vuoto 0 (il tipo che non ha abitanti), e la regola ex falso quodlibet, che permette di derivare qualsiasi proposizione da $\perp$, corrisponde a una funzione dal tipo vuoto a qualsiasi tipo, che è vacuamente vera poiché non ci sono elementi del tipo vuoto a cui applicarla.

Nel contesto del calcolo di Lambek e della sintassi categoriale, la corrispondenza di Curry-Howard assume una forma particolarmente ricca. Le categorie sintattiche corrispondono a tipi, le derivazioni sintattiche corrispondono a termini lambda tipizzati, e la combinazione sintattica corrisponde alla composizione di funzioni o all'applicazione di funzione. Lo slash destro A/B corrisponde al tipo funzionale $B \rightarrow A$, e una derivazione che combina un'espressione di categoria A/B con un'espressione di categoria B per ottenere A corrisponde all'applicazione di una funzione di tipo $B \rightarrow A$ a un argomento di tipo B .

La regola di introduzione dello slash nel calcolo di Lambek, che permette di derivare A/B da una derivazione ipotetica di A da B , corrisponde all'astrazione lambda. Questa connessione rivela che il calcolo di Lambek può essere visto come un frammento della logica intuizionista non commutativa, e che le derivazioni sintattiche portano contenuto computazionale esattamente come le dimostrazioni logiche.

Un'estensione importante della corrispondenza di Curry-Howard riguarda la logica di ordine superiore e i sistemi di tipi polimorfi. Il sistema F di Girard, che estende il lambda calcolo con quantificazione universale sui tipi, corrisponde alla logica proposizionale intuizionista del secondo ordine. In questo sistema, una formula $\forall \alpha. A(\alpha)$ corrisponde a un tipo polimorfo $\forall \alpha. T(\alpha)$, e una dimostrazione di tale formula corrisponde a un termine che funziona uniformemente per tutti i tipi. Questo ha applicazioni nella semantica del linguaggio naturale per modellare fenomeni che richiedono polimorfismo, come certi tipi di coordinazione e quantificazione generalizzata.

La corrispondenza si estende anche a logiche modali e substructurali, rilevanti per la modellazione di fenomeni linguistici avanzati. La logica lineare di Girard, in cui le assunzioni devono essere usate esattamente una volta, corrisponde a sistemi di tipi lineari dove le variabili devono essere usate esattamente una volta. Questo ha applicazioni nella semantica di fenomeni come la quantificazione e le espressioni anaforiche, dove la linearità cattura requisiti di uso delle risorse.

Il lambda calcolo con tipi dipendenti, corrispondente alla teoria dei tipi di Martin-Löf, permette che i tipi dipendano da valori. Questo sistema corrisponde alla logica predicativa intuizionista, dove le formule possono contenere quantificatori su individui. Nel contesto linguistico, i tipi dipendenti possono modellare fenomeni di anafora dinamica, dove il tipo di un'espressione successiva dipende dal valore introdotto da un'espressione precedente.

La corrispondenza di Curry-Howard non è solo un isomorfismo astratto, ma ha conseguenze computazionali concrete. La normalizzazione nel lambda calcolo, il processo di ridurre un termine alla sua forma normale attraverso beta-riduzioni successive, corrisponde all'eliminazione del taglio nella teoria della dimostrazione, il processo di trasformare una dimostrazione che usa lemmi intermedi in una dimostrazione diretta. Il teorema di normalizzazione forte, che afferma che ogni sequenza di riduzioni termina in una forma normale unica, garantisce che la computazione semantica è ben definita e produce sempre un risultato.

Questa connessione tra dimostrazione e computazione ha ispirato lo sviluppo di assistenti alla dimostrazione e linguaggi di programmazione basati sulla teoria dei tipi dipendenti, come Coq, Agda e Idris. Questi sistemi permettono di scrivere programmi la cui correttezza è garantita da dimostrazioni formali incorporate nel codice stesso. Nel contesto della linguistica computazionale, questi strumenti possono essere usati per implementare grammatiche formali con garanzie di correttezza verificate meccanicamente.

La corrispondenza di Curry-Howard fornisce anche una prospettiva unificata sull'interfaccia sintassi-semantica. Tradizionalmente, la sintassi era vista come un sistema di regole per generare strutture, mentre la semantica era vista come un sistema di interpretazione di tali strutture. La corrispondenza rivela che queste due dimensioni sono in realtà due facce della stessa medaglia: una derivazione sintattica nel calcolo di Lambek è simultaneamente una dimostrazione logica e un programma che computa il significato. Questa dualità elimina la necessità di postulare regole di interpretazione separate: la struttura della dimostrazione determina automaticamente la computazione del significato.

Nell'ambito delle grammatiche categoriali combinatorie, la corrispondenza di Curry-Howard si manifesta nell'isomorfismo tra regole combinatorie sintattiche e combinatori nel lambda

calcolo. La regola di composizione funzionale, per esempio, corrisponde al combinatore B di logica combinatoria, definito come $B = \lambda f.\lambda g.\lambda x.f(g\ x)$. Questa connessione permette di interpretare le derivazioni CCG come termini nel lambda calcolo combinatorio, fornendo una semantica computazionale diretta.

Type-Logical Grammars e Architetture Multimodali

Le grammatiche type-logical rappresentano una famiglia di formalismi grammaticali che estendono e generalizzano il calcolo di Lambek originale in varie direzioni, mantenendo l'idea fondamentale che la grammatica può essere concepita come un sistema di tipi logico. Questi formalismi cercano di catturare una gamma più ampia di fenomeni linguistici introducendo modalità, operatori strutturali controllati, e architetture multimodali che permettono diversi regimi di combinazione sintattica all'interno dello stesso sistema grammaticale.

Un'innovazione chiave nelle type-logical grammars è l'introduzione di operatori modali che controllano le proprietà strutturali delle derivazioni. Nel calcolo di Lambek puro, la struttura delle sequenze è rigidamente non commutativa e associativa. Tuttavia, molti fenomeni linguistici sembrano richiedere flessibilità selettiva in queste proprietà strutturali. Per esempio, la coordinazione sembra richiedere una certa commutatività locale, mentre fenomeni di scrambling in lingue come il tedesco o il giapponese sembrano permettere riordinamenti controllati dei costituenti.

I calcoli di Lambek multimodali affrontano questa sfida introducendo una famiglia di operatori di prodotto e slash indicizzati da modalità. Ogni modalità m è associata a un insieme di proprietà strutturali (commutatività, associatività, contrazione, indebolimento) che specificano come le espressioni possono essere riorganizzate quando sono combinate usando operatori di quella modalità. Per esempio, potremmo avere un prodotto $\cdot_1$ che è strettamente ordinato e non permette permutazioni, e un prodotto $\cdot_2$ che permette commutatività locale per modellare coordinazione simmetrica.

Formalmente, un calcolo di Lambek multimodale è specificato attraverso un insieme di modalità M e una relazione di accessibilità tra modalità che determina quando regole strutturali possono essere applicate. Ogni modalità m viene con i suoi operatori di prodotto $\cdot_m$ e slash $/_m$ e $\backslash_m$, e le regole strutturali sono controllate da postulati che specificano le proprietà di ciascuna modalità. Questo framework permette di catturare variazione cross-linguistica nell'ordine delle parole e nella struttura sintattica semplicemente variando l'insieme di modalità e i postulati strutturali.

Un esempio influente di grammatica multimodale è il sistema proposto da Glyn Morrill e collaboratori, che usa modalità per distinguere tra diversi tipi di costituenza sintattica. Una modalità potrebbe governare la combinazione di teste lessicali con i loro complementi, richiedendo ordine rigido, mentre un'altra modalità potrebbe governare l'aggiunta di modificatori, permettendo maggiore flessibilità nell'ordinamento. Questo approccio cattura l'osservazione empirica che complementi e modificatori si comportano diversamente rispetto a fenomeni come l'estrazione e la coordinazione.

Le grammatiche type-logical hanno anche sviluppato meccanismi sofisticati per trattare le dipendenze a lunga distanza e i fenomeni di isola. Questi fenomeni, che hanno motivato l'introduzione di tracce e movimenti nella tradizione generativa, possono essere modellati nelle type-logical grammars attraverso l'uso di modalità estrattive e operatori di chiusura. L'idea è che un'espressione che contiene un "buco" (una posizione non saturata) porta un'annotazione modale che tiene traccia della natura di quel buco e delle restrizioni sulla sua saturazione.

Per esempio, in una frase con estrazione come "Quale libro pensi che Maria abbia letto?", il sintagma interrogativo "quale libro" può essere analizzato come avente una categoria che cerca una frase con un buco di tipo NP nella posizione dell'oggetto. Questa richiesta è codificata attraverso una modalità estrattiva $\Diamond$, e la derivazione deve garantire che il buco sia eventualmente saturato nella posizione appropriata. Le restrizioni sulle isole possono essere codificate limitando dove e come le modalità estrattive possono propagarsi attraverso la struttura sintattica.

Michael Moortgat ha sviluppato un calcolo di Lambek multimodale arricchito con operatori modali unari che permettono di modellare una varietà di fenomeni sintattici complessi. In questo sistema, oltre ai prodotti e agli slash binari, abbiamo operatori unari $\Diamond_m$ e $\Box_m$ per ciascuna modalità m , che possono essere usati per "impacchettare" categorie e controllare la loro distribuzione sintattica. Questi operatori soddisfano leggi di aggiunzione simili a quelle della logica modale, creando un ponte tra la sintassi categoriale e la semantica dei mondi possibili.

Un'applicazione particolarmente elegante delle modalità riguarda il trattamento della quantificazione e dello scope. La quantificazione è nota per generare ambiguità di scope: la frase "Ogni studente ha letto qualche libro" può significare che c'è una potenziale differenza nel libro per ciascuno studente, oppure che c'è un libro specifico che tutti hanno letto. Nelle type-logical grammars, queste letture possono essere derivate usando modalità che controllano dove e quando i quantificatori possono prendere scope, evitando la necessità di postulare movimenti sintattici astratti o livelli di rappresentazione multipli.

La semantica delle grammatiche multimodali richiede modelli matematici più ricchi rispetto ai semplici modelli categoriali del calcolo di Lambek standard. Tipicamente, si usano modelli in categorie residue multimodali, dove ciascuna modalità è interpretata come un diverso prodotto tensoriale con i suoi residui associati. La soundness e completeness rispetto a questi modelli garantisce che le derivazioni sintattiche abbiano interpretazioni semantiche ben definite.

Un'altra linea di sviluppo nelle type-logical grammars riguarda l'integrazione di informazione prosodica e fonologica. Richard Oehrle e altri hanno proposto architetture multimodali in cui diverse dimensioni della struttura linguistica (sintassi, prosodia, informazione) sono rappresentate in domini modali separati ma interconnessi. In questo approccio, un'espressione linguistica non ha solo una categoria sintattica ma anche una categoria prosodica e una categoria informativa, e le regole di combinazione devono essere sensibili a tutte queste dimensioni simultaneamente.

Le grammatiche di dipendenza lessicale (Lexical-Functional Grammars, LFG) e le grammatiche a struttura sintagmatica guidata dalle teste (Head-Driven Phrase Structure

Grammar, HPSG) hanno anche influenzato lo sviluppo di type-logical grammars ibride che cercano di combinare i vantaggi del lessicalismo e della composizionalità rigorosa con la capacità di modellare fenomeni come l'accordo e il controllo. In questi sistemi, le categorie sintattiche sono arricchite con features strutturate che codificano informazioni morfosintattiche, e le regole combinatorie devono garantire la compatibilità delle features oltre alla corretta combinazione categoriale.

Dependency Parsing come Teoria dei Grafi

Il dependency parsing rappresenta un paradigma alternativo alla costituenza sintattica per modellare la struttura delle frasi, basato sull'idea che la sintassi è fondamentalmente una questione di relazioni binarie asimmetriche tra parole. In una rappresentazione a dipendenze, ciascuna parola nella frase (eccetto una, la radice) è connessa esattamente a un'altra parola tramite una relazione di dipendenza direzionale. La parola da cui dipende un'altra parola è chiamata testa (head), mentre la parola dipendente è chiamata modificatore o complemento. L'insieme di tutte queste relazioni forma un albero di dipendenza che cattura la struttura grammaticale della frase.

Dal punto di vista della teoria dei grafi, un albero di dipendenza per una frase di n parole è un grafo orientato $G = (V, E)$ dove V è l'insieme dei nodi corrispondenti alle parole della frase (più eventualmente un nodo radice artificiale), ed E è l'insieme degli archi orientati che rappresentano le dipendenze. Le proprietà fondamentali che un valido albero di dipendenza deve soddisfare sono: (1) esiste esattamente un nodo radice che non ha archi entranti; (2) ogni altro nodo ha esattamente un arco entrante; (3) esiste un cammino unico dalla radice a ogni altro nodo; (4) il grafo è aciclico; (5) gli archi non si incrociano (nel caso di dipendenze proiettive).

La condizione di non incrocio (projectivity) è particolarmente importante dal punto di vista computazionale. Un albero di dipendenza è detto proiettivo se, per ogni arco dalla parola i alla parola j , tutte le parole tra i e j (nell'ordine lineare della frase) sono discendenti di i nell'albero. Graficamente, questo significa che quando le parole sono disposte in ordine lineare e gli archi sono disegnati sopra o sotto la linea, nessun arco incrocia un altro. La maggior parte delle frasi nella maggior parte delle lingue ammette analisi proiettive, ma esistono costruzioni, particolarmente in lingue con ordine delle parole relativamente libero, che richiedono dipendenze non proiettive.

Dal punto di vista formale, il problema del dependency parsing può essere formulato come un problema di ottimizzazione su grafi. Dato un insieme di possibili dipendenze con pesi associati (derivati tipicamente da un modello statistico), l'obiettivo è trovare l'albero di dipendenza di peso massimo che soddisfi i vincoli strutturali. Per alberi proiettivi, questo problema può essere risolto efficientemente usando algoritmi di programmazione dinamica come l'algoritmo di Eisner, che ha complessità temporale $O(n^3)$ dove n è la lunghezza della frase.

L'algoritmo di Eisner costruisce l'albero di dipendenza ottimale combinando ricorsivamente sotto-alberi più piccoli. L'idea chiave è rappresentare i sotto-problemi attraverso "span" contigue di parole e distinguere tra span "complete" (in cui la testa ha già raccolto tutti i suoi dipendenti da un lato) e span "incomplete" (in cui la testa sta ancora cercando dipendenti).

La tabella di programmazione dinamica memorizza i punteggi migliori per ogni tipo di span, e la ricorsione combina span più piccoli in span più grandi rispettando i vincoli di proiettività.

Per le dipendenze non proiettive, il problema diventa più complesso dal punto di vista computazionale. Il problema di trovare l'albero di dipendenza non proiettivo di peso massimo è NP-hard in generale, sebbene esistano algoritmi di approssimazione e euristiche che funzionano bene in pratica. Un approccio influente è l'algoritmo di Chu-Liu-Edmonds per trovare il maximum spanning tree (MST) in un grafo orientato pesato. Questo algoritmo può essere adattato al dependency parsing considerando tutte le possibili dipendenze tra parole come archi di un grafo completo pesato, e trovando l'albero ricoprente di peso massimo radicato nel nodo radice.

L'algoritmo di Chu-Liu-Edmonds opera identificando cicli nel grafo e contraendoli in super-nodi, ricorsivamente risolvendo il problema sul grafo contratto, e poi espandendo la soluzione rimuovendo un arco appropriato dal ciclo. La complessità temporale è $O(n^2)$ per grafi densi, rendendolo computazionalmente trattabile anche per frasi relativamente lunghe. Un vantaggio di questo approccio è che garantisce di trovare l'albero ottimale globale, a differenza degli approcci greedy o basati su transizioni che possono rimanere intrappolati in ottimi locali.

I parser basati su transizioni rappresentano un'altra classe importante di algoritmi per il dependency parsing, ispirati ai sistemi di shift-reduce usati nel parsing di grammatiche context-free. Questi parser mantengono una configurazione composta da uno stack, un buffer di parole da processare, e un insieme di archi di dipendenza già costruiti. Il parsing procede applicando una sequenza di transizioni che manipolano questa configurazione fino a raggiungere uno stato finale in cui tutte le parole sono state processate e tutte le dipendenze sono state stabilite.

Le transizioni standard in un sistema shift-reduce per dependency parsing includono: (1) SHIFT, che sposta la prima parola dal buffer sullo stack; (2) LEFT-ARC(label), che crea una dipendenza etichettata dal secondo elemento dello stack al primo, e rimuove il primo dallo stack; (3) RIGHT-ARC(label), che crea una dipendenza dal primo elemento dello stack al secondo, e rimuove il secondo. Diverse varianti di questi sistemi sono state proposte, includendo transizioni aggiuntive per gestire dipendenze non proiettive o per migliorare l'efficienza.

Il vantaggio principale dei parser basati su transizioni è la loro efficienza computazionale: il parsing può essere completato in tempo lineare $O(n)$ se assumiamo tempo costante per decidere quale transizione applicare. In pratica, la decisione su quale transizione applicare viene presa usando un classificatore di machine learning addestrato su un corpus annotato (treebank). I classificatori possono basarsi su una varietà di features estratte dalla configurazione corrente, includendo le parole sullo stack e nel buffer, le loro parti del discorso, le dipendenze già costruite, e informazioni morfosintattiche.

Negli ultimi anni, l'avvento delle reti neurali ha rivoluzionato il dependency parsing. I parser neurali usano rappresentazioni distribuite (embeddings) delle parole e delle features, e reti neurali profonde per predire le transizioni o per assegnare punteggi alle possibili dipendenze. Un approccio influente è il parser neurale basato su transizioni di Chen e

Manning, che usa una rete neurale feed-forward con uno strato nascosto per predire la transizione ottimale basandosi sugli embeddings delle parole sullo stack e nel buffer.

Più recentemente, i modelli basati su attenzione e transformer hanno ulteriormente migliorato le prestazioni del dependency parsing. Questi modelli possono catturare dipendenze a lungo raggio e interazioni complesse tra parole attraverso meccanismi di self-attention che permettono a ogni parola di "guardare" a tutte le altre parole nella frase. I parser basati su BERT e altri modelli di linguaggio pre-addestrati hanno raggiunto accuratissime eccezionali su benchmark standard, tipicamente superiori al 95% di Unlabeled Attachment Score (UAS) per l'inglese.

Dal punto di vista teorico, esiste una connessione profonda tra dependency parsing e parsing con grammatiche categoriali. Ogni grammatica categoriale definisce implicitamente una struttura di dipendenza attraverso la nozione di applicazione funzionale: quando un funtore si applica a un argomento, l'argomento dipende dalla testa del funtore. Questa corrispondenza è stata esplorata formalmente da vari ricercatori, mostrando che le derivazioni nel calcolo di Lambek possono essere tradotte in alberi di dipendenza e viceversa, sotto opportune assunzioni.

In particolare, le grammatiche categoriali lessicalizzate, in cui ogni categoria è associata a una specifica parola lessicale, possono essere viste come grammatiche di dipendenza con un sistema ricco di etichette di dipendenza. Ogni categoria categoriale specifica non solo che tipo di dipendente una testa cerca, ma anche la direzione e l'ordine in cui questi dipendenti devono essere saturati. Questo porta a una corrispondenza stretta tra derivazioni categoriali e alberi di dipendenza, con la categoria categoriale di una parola che determina la sua funzione sintattica e le sue dipendenze.

La teoria dei grafi fornisce anche strumenti per analizzare proprietà globali degli alberi di dipendenza e delle grammatiche che li generano. Per esempio, la nozione di "centrality" in teoria dei grafi può essere usata per identificare le parole sintatticamente più centrali in una frase, che spesso corrispondono ai predicati principali. La betweenness centrality di un nodo misura quanti cammini più brevi tra coppie di altri nodi passano attraverso quel nodo, fornendo una misura della sua importanza strutturale.

Un altro concetto rilevante è quello di treewidth, che misura quanto un grafo è "simile a un albero". Gli alberi di dipendenza per loro natura hanno treewidth 1, ma se consideriamo rappresentazioni più generali che permettono archi multipli o strutture di grafici aciclici diretti, la treewidth diventa una misura della complessità strutturale. Grammatiche con treewidth limitato possono essere parsate più efficientemente, e esistono risultati teorici che caratterizzano la complessità del parsing in funzione della treewidth della grammatica.

Le rappresentazioni a dipendenza hanno anche il vantaggio di essere relativamente neutrali rispetto alla tipologia linguistica, a differenza delle grammatiche a costituenza che possono richiedere assunzioni specifiche sulla struttura delle frasi. Una dipendenza binaria tra una testa e un dipendente è un concetto sufficientemente generale da poter essere applicato a lingue con ordini delle parole molto diversi, da lingue rigidamente configurazionali come l'inglese a lingue con ordine relativamente libero come il russo o il latino.

Tuttavia, le rappresentazioni a dipendenza non sono senza controversie. Una questione teorica importante riguarda la scelta della testa in costruzioni con funzione grammaticale primaria non ovvia. Per esempio, nelle costruzioni copulari come "Maria è medico", è controverso se la testa sia il verbo copula "è" o il predicato nominale "medico". Diverse convenzioni di annotazione fanno scelte diverse su questi casi, riflettendo diverse assunzioni teoriche sulla natura della struttura sintattica.

Un'altra questione riguarda il trattamento della coordinazione. Nelle grammatiche a dipendenza, la coordinazione pone sfide particolari perché gli elementi coordinati sono tipicamente trattati asimmetricamente, con uno che dipende dall'altro o entrambi che dipendono dalla congiunzione. Questo contrasta con l'intuizione semantica che i coordinati dovrebbero essere trattati simmetricamente. Diverse soluzioni sono state proposte, includendo l'uso di dipendenze speciali per la coordinazione o l'adozione di rappresentazioni a grafo più generali che permettono strutture non-albero.

Algebre di Alberi Sintattici e Operazioni Composizionali

Le algebre di alberi sintattici forniscono un framework matematico per modellare la struttura sintattica come insieme di oggetti algebrici con operazioni ben definite. Questo approccio, sviluppato principalmente nel contesto delle Tree Adjoining Grammars (TAG) e dei loro successori, concepisce la costruzione sintattica come processo di combinazione di alberi elementari attraverso operazioni specifiche. L'intuizione fondamentale è che gli alberi sintattici non sono semplicemente strutture dati passive, ma oggetti matematici attivi che possono essere composti e trasformati secondo regole algebriche.

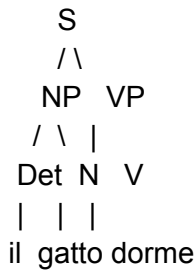
Una Tree Adjoining Grammar è specificata da un insieme finito di alberi elementari di due tipi: alberi iniziali (initial trees) e alberi ausiliari (auxiliary trees). Gli alberi iniziali rappresentano strutture sintattiche complete minimali, mentre gli alberi ausiliari rappresentano modificatori o materiale ricorsivo che può essere inserito all'interno di altri alberi. Un albero ausiliario è caratterizzato dalla presenza di un nodo speciale chiamato foot node, dello stesso tipo categoriale del nodo radice, che indica dove l'albero può essere agganciato (adjoined) ad altri alberi.

Le due operazioni fondamentali nelle TAG sono la sostituzione (substitution) e l'aggiunta (adjunction). La sostituzione rimpiazza un nodo foglia in un albero con un albero iniziale che ha la stessa etichetta categoriale di quel nodo. L'aggiunta inserisce un albero ausiliario in un nodo interno di un altro albero: il sottoalbero radicato in quel nodo viene rimosso e riattaccato al foot node dell'albero ausiliario, mentre l'albero ausiliario stesso viene innestato nella posizione originale. Queste operazioni catturano rispettivamente la saturazione di argomenti obbligatori e l'aggiunta di modificatori opzionali.

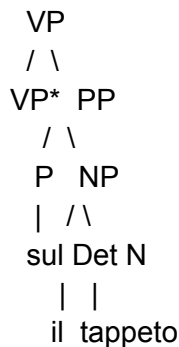
Formalmente, possiamo definire un'algebra di alberi come una struttura $(T, \{op_1, op_2, \dots\})$ dove T è un insieme di alberi sintattici e $\{op_1, op_2, \dots\}$ è un insieme di operazioni che prendono alberi come input e producono alberi come output. Nel caso delle TAG, le operazioni principali sono sostituzione e aggiunta, ma altre operazioni possono essere definite per estendere il potere espressivo del formalismo.

La composizione di alberi attraverso queste operazioni può essere formalizzata usando la teoria delle algebre universali. Un'algebra di alberi soddisfa determinate leggi algebriche, come l'associatività di certe operazioni, che garantiscono che il risultato della composizione è ben definito indipendentemente dall'ordine in cui le operazioni sono applicate (quando questo è permesso). Queste proprietà algebriche hanno conseguenze importanti per la computabilità e la complessità del parsing.

Un esempio concreto aiuta a illustrare come funzionano queste operazioni. Consideriamo la frase "Il gatto dorme sul tappeto". Potremmo avere un albero iniziale per "il gatto dorme" che ha la struttura:



E un albero ausiliario per "sul tappeto" che modifica il verbo:



Il nodo VP* con asterisco è il foot node. L'operazione di adjunction inserisce questo albero ausiliario nel nodo VP dell'albero iniziale, producendo l'albero completo per l'intera frase. Il sottoalbero VP originale (contenente solo il verbo "dorme") viene spostato nella posizione del foot node, risultando in una struttura che cattura correttamente la modificazione avverbiale.

Le TAG hanno diverse proprietà formali importanti. Generano un insieme di linguaggi che sono mildly context-sensitive, cioè che vanno oltre le grammatiche context-free ma rimangono all'interno di una classe trattabile computazionalmente. Specificamente, le TAG possono catturare dipendenze a lunga distanza e accordi cross-seriali che sono oltre il potere delle CFG, ma il problema del parsing rimane polinomiale ($O(n^6)$ per l'algoritmo standard, riducibile a $O(n^3)$ per grammatiche lessicalizzate).

La proprietà di mild context-sensitivity è caratterizzata formalmente da quattro condizioni: (1) il riconoscimento è polinomiale; (2) i linguaggi generati sono semi-lineari (le funzioni di crescita sono polinomiali); (3) il linguaggio può rappresentare dipendenze incrociate limitate;

(4) ogni linguaggio ha la proprietà di pumping costante. Queste proprietà rendono le TAG attraenti dal punto di vista computazionale pur mantenendo sufficiente potere espressivo per molti fenomeni sintattici del linguaggio naturale.

Le TAG lessicalizzate (LTAG), introdotte da Aravind Joshi e Yves Schabes, richiedono che ogni albero elementare contenga almeno un'ancora lessicale (lexical anchor), cioè una parola effettiva. Questa restrizione ha due vantaggi importanti: (1) cattura l'idea che la struttura sintattica è proiettata dal lessico; (2) riduce la complessità del parsing perché il numero di alberi da considerare in ogni punto è limitato dalle parole effettivamente presenti nella frase. Le LTAG hanno dimostrato di essere particolarmente efficaci per modellare la valenza verbale e le costruzioni lessicalmente ancorate.

La semantica delle TAG può essere specificata attraverso una derivazione sincrona con la semantica, simile all'approccio nelle grammatiche categoriali. Ogni albero elementare è associato a una rappresentazione semantica, e le operazioni di sostituzione e adjunction sono accompagnate da operazioni di composizione semantica corrispondenti. Per esempio, la sostituzione può corrispondere all'applicazione di funzione, mentre l'adjunction può corrispondere alla modifica di una funzione o all'introduzione di un quantificatore.

Una estensione importante delle TAG classiche sono le Multi-Component TAG (MCTAG), che permettono di raggruppare più alberi elementari in "set trees" che devono essere utilizzati insieme. Questo è utile per modellare fenomeni come i verbi a supporto o costruzioni idiomatiche che coinvolgono elementi lessicali multipli ma sintatticamente separati. Le MCTAG mantengono le proprietà di mild context-sensitivity sotto certe restrizioni sulla composizione dei tree sets.

Le Synchronous TAG (STAG), sviluppate per la traduzione automatica e la generazione di linguaggio naturale, operano simultaneamente su coppie di alberi in due linguaggi (per esempio, due lingue naturali diverse, o sintassi e semantica). Le operazioni di sostituzione e adjunction sono sincronizzate tra i due alberi, garantendo che la struttura sintattica nella lingua sorgente corrisponda appropriatamente alla struttura nella lingua target. Questo framework ha trovato applicazioni nella traduzione automatica basata su strutture sintattiche.

Dal punto di vista algebrico, le operazioni delle TAG possono essere viste come morfismi in una categoria appropriata di alberi. Gli alberi formano gli oggetti della categoria, e le operazioni di composizione formano i morfismi. La composizione di morfismi corrisponde alla composizione sequenziale di operazioni di costruzione di alberi. Questa prospettiva categoriale fornisce un framework unificante per comprendere diverse operazioni su alberi e le loro proprietà composizionali.

Un'altra linea di ricerca ha esplorato l'uso di automi su alberi (tree automata) per riconoscere e generare linguaggi di alberi sintattici. Un automa su alberi opera su alberi piuttosto che su stringhe, processando l'albero in modo bottom-up o top-down e accettando o rifiutando basandosi sugli stati raggiunti. Gli automi su alberi forniscono un framework computazionale per ragionare su insiemi di alberi sintattici e sulle loro proprietà, e sono stati usati per studiare la complessità di varie classi di grammatiche.

Le algebre di termini, studiate in logica matematica e nell'informatica teorica, forniscono un altro framework correlato per modellare la struttura sintattica. In questo approccio, gli alberi

sintattici sono rappresentati come termini in un linguaggio formale, con funzioni che corrispondono a nodi dell'albero e argomenti che corrispondono a sottoalberi. Le operazioni su termini, come la sostituzione e l'unificazione, possono essere usate per modellare la costruzione e la trasformazione sintattica.

La riscrittura di termini (term rewriting) è una tecnica per specificare trasformazioni su alberi sintattici attraverso regole di riscrittura. Una regola di riscrittura $l \rightarrow r$ specifica che un termine che corrisponde al pattern l può essere trasformato nel termine r . Sistemi di riscrittura di termini sono stati usati per modellare fenomeni come le trasformazioni sintattiche nella grammatica generativa, con il vantaggio di fornire una semantica operativa chiara e proprietà formali verificabili come la confluenza e la terminazione.

Fenomeni Avanzati: Quantificazione, Anafora e Dipendenze a Lunga Distanza

I fenomeni linguistici avanzati come la quantificazione generalizzata, l'anafora dinamica e le dipendenze a lunga distanza hanno motivato estensioni sofisticate dei framework categoriali di base, portando allo sviluppo di sistemi formali che integrano logica, teoria dei tipi e teoria delle categorie in modi sempre più raffinati. Questi fenomeni pongono sfide particolari per il principio di composizionalità e richiedono meccanismi che vanno oltre la semplice applicazione funzionale.

La quantificazione generalizzata, studiata sistematicamente da Barwise e Cooper negli anni Ottanta, tratta i sintagmi nominali quantificati non come termini che denotano individui, ma come quantificatori generalizzati che denotano relazioni tra insiemi (o equivalentemente, proprietà di proprietà). Un quantificatore generalizzato è una funzione che prende due proprietà P e Q e restituisce un valore di verità. Per esempio, il quantificatore "ogni" denota la relazione $\lambda P.\lambda Q.\forall x[P(x) \rightarrow Q(x)]$, che è vera quando l'insieme degli oggetti con proprietà Q include l'insieme degli oggetti con proprietà P .

Questa analisi cattura elegantemente vari fenomeni quantificazionali. I quantificatori monotoni crescenti come "qualche" hanno la proprietà che se "qualche F è G " è vero e G' include G , allora "qualche F è G' " è anche vero. I quantificatori monotoni decrescenti come "nessuno" hanno la proprietà duale. Queste proprietà di monotonicità hanno conseguenze importanti per il ragionamento inferenziale e possono essere caratterizzate formalmente in termini di teoria degli ordini.

Tuttavia, la quantificazione generalizzata introduce un problema per la composizionalità: come dovremmo analizzare frasi con quantificatori multipli come "Ogni studente ha letto qualche libro"? Questa frase ha due letture a seconda di quale quantificatore prende scope su quale: nella lettura de dicto, per ogni studente esiste un libro (potenzialmente diverso) che quello studente ha letto; nella lettura de re, esiste un libro specifico tale che ogni studente lo ha letto. La semantica compositiva deve spiegare come entrambe le letture emergono dalla stessa struttura sintattica superficiale.

Le grammatiche categoriali hanno sviluppato vari meccanismi per trattare l'ambiguità di scope. Un approccio, sviluppato da Hendriks e altri, usa il type-raising e la composizione

funzionale per derivare diverse strutture che corrispondono a diverse relazioni di scope. Il type-raising permette a un sintagma nominale quantificato di combinare con il predicato in modi diversi, generando letture di scope diverse. Un altro approccio usa modalità speciali o operatori di scope-shifting che permettono ai quantificatori di "muoversi" a posizioni di scope più alte mantenendo la composizionalità.

La logica dinamica, sviluppata da Groenendijk e Stokhof, fornisce un framework alternativo per trattare la quantificazione e l'anafora. In questo approccio, il significato di una frase non è semplicemente un valore di verità o una proposizione, ma un programma che aggiorna lo stato informativo del discorso. L'anafora pronominale è trattata come riferimento a entità introdotte in aggiornamenti precedenti, piuttosto che come binding statico di variabili.

Un esempio classico che motiva la semantica dinamica è il donkey sentence: "Ogni contadino che possiede un asino lo batte". Il pronome "lo" sembra riferirsi all'asino introdotto nella relativa, ma questo asino è nell'ambito di un quantificatore esistenziale, rendendo problematico il binding tradizionale. Nella semantica dinamica, il quantificatore "un asino" introduce un discourse referent che rimane accessibile per la pronominalizzazione successiva, anche attraverso confini quantificazionali.

La Dynamic Predicate Logic (DPL), formalizzata da Groenendijk e Stokhof, modella questo comportamento trattando le formule come relazioni tra assegnamenti di variabili (assignment functions). Una formula φ denota una relazione R tale che $(g, h) \in R$ se e solo se φ è vero nell'assegnamento g e h è un assegnamento che può differire da g sui valori di variabili introdotte da φ . La composizione di formule corrisponde alla composizione relazionale di queste relazioni, catturando il flusso di informazione attraverso il discorso.

La semantica dinamica può essere integrata con la grammatica categoriale attraverso l'uso di tipi dinamici. Invece di interpretare le espressioni come funzioni statiche, le interpretiamo come relazioni tra contesti di input e output. Un nome proprio in semantica dinamica non denota semplicemente un individuo, ma una funzione che prende un contesto e restituisce lo stesso contesto arricchito con l'informazione che quell'individuo è rilevante. I quantificatori manipolano questi contesti in modi specifici, introducendo o restringendo discourse referents.

Le dipendenze a lunga distanza, come quelle che occorrono nelle domande *wh* e nelle frasi relative, pongono un altro tipo di sfida. In "Quale libro pensi che Maria abbia letto?", esiste una dipendenza tra il sintagma interrogativo in posizione iniziale e la posizione dell'oggetto del verbo embedded "letto". Nelle grammatiche trasformazionali, questo è modellato attraverso movimento, ma nelle grammatiche categoriali dobbiamo trovare meccanismi compositivi che permettano questa dipendenza senza ricorrere a trasformazioni.

Un approccio, già menzionato, usa il type-raising e la composizione funzionale per permettere al sintagma *wh* di combinare con materiale che contiene un "buco" (gap) sintattico. Il buco è rappresentato attraverso una categoria particolare, spesso notata con uno slash che cerca il tipo mancante. Per esempio, "pensi che Maria abbia letto" può ricevere la categoria *S/NP*, indicando che cerca un oggetto *NP* per formare una frase completa. Il sintagma interrogativo, con categoria appropriata, può quindi saturare questo buco.

Un'altra linea di ricerca usa la continuation semantics per modellare le dipendenze a lunga distanza. In questo approccio, influenzato dalla semantica delle continuations nella programmazione funzionale, le espressioni sono interpretate non come denotazioni dirette ma come funzioni che prendono una continuazione (una funzione che rappresenta "il resto della computazione") e producono un risultato. Le dipendenze a lunga distanza emergono attraverso manipolazione di continuazioni, permettendo a un elemento di "saltare" intervening material e connettersi direttamente con la sua posizione di interpretazione.

Le monadi, concetto centrale nella teoria delle categorie e nella programmazione funzionale, forniscono un framework astratto per modellare effetti computazionali nella semantica del linguaggio naturale. Una monade è una tripla (M, η, μ) dove M è un endofunttore su una categoria, η è una trasformazione naturale dall'identità a M (chiamata unit), e μ è una trasformazione naturale da $M \circ M$ a M (chiamata join), soddisfacenti determinate leggi di coerenza. Nel contesto della semantica, le monadi permettono di incorporare effetti come il non-determinismo, lo stato, o l'intensionalità mantenendo la composizionalità.

La monade delle continuazioni, per esempio, è particolarmente utile per modellare la quantificazione e le dipendenze a lunga distanza. Se α è il tipo delle risposte (answer type), tipicamente t per valori di verità, allora la monade delle continuazioni trasforma ogni tipo a nel tipo $(a \rightarrow \alpha) \rightarrow \alpha$. Un'espressione di tipo a è interpretata come una funzione che prende una continuazione (cosa fare con un valore di tipo a) e produce una risposta. I quantificatori si interpretano naturalmente in questo framework: "ogni F " denota una funzione che prende una continuazione k e restituisce vero se k restituisce vero per tutti gli F .

La teoria dei tipi di Martin-Löf e sistemi correlati offrono un framework ancora più ricco per la semantica formale, permettendo di esprimere dipendenze non solo funzionali ma anche proposizionali. In un sistema con tipi dipendenti, possiamo avere tipi $\prod(x:A).B(x)$ (prodotto dipendente) e $\sum(x:A).B(x)$ (somma dipendente) dove il tipo B dipende dal valore di x . Questo permette di esprimere quantificazione universale ed esistenziale a livello dei tipi stessi, fornendo una fondazione logica più espressiva per la semantica.

Un'applicazione dei tipi dipendenti alla linguistica riguarda il trattamento dell'anafora. Un pronome può essere interpretato come una variabile dipendente il cui tipo dipende dal suo antecedente. Se x è di tipo "persona", allora un pronome che si riferisce a x può avere tipo dipendente $\text{Pron}(x)$, garantendo che il pronome può essere usato solo in contesti dove x è disponibile. Questo cattura formalmente le restrizioni di binding che governano l'uso dei pronomi senza ricorrere a meccanismi stipulativi.

I record types, sviluppati da Robin Cooper nella Type Theory with Records (TTR), forniscono un altro strumento per modellare fenomeni semantici complessi. Un record type è una struttura che specifica una collezione di campi etichettati con i loro tipi, permettendo di rappresentare informazione strutturata in modo flessibile. Questo è utile per modellare fenomeni come la modificazione graduale di informazione, l'accordo morfologico, e le relazioni anaforiche che coinvolgono informazione parziale.

Le isole sintattiche, fenomeno studiato estensivamente nella grammatica generativa, rappresentano contesti da cui l'estrazione è impossibile o degradata. Per esempio, "Di quale libro hai incontrato l'autore?" è marginale o inaccettabile in molti dialetti, anche se "Quale libro hai letto?" è perfettamente accettabile. Le grammatiche categoriali hanno sviluppato

vari meccanismi per catturare questi effetti di isola, spesso attraverso restrizioni sulle modalità estrattive o sulla propagazione di gap.

Michael Moortgat ha proposto un trattamento delle isole basato sulla logica multimodale, dove diverse modalità controllano diversi tipi di dipendenze. Le isole sono modellate come contesti che bloccano la propagazione di certe modalità estrattive. Per esempio, un'isola complessa come una frase relativa potrebbe permettere dipendenze locali ma bloccare dipendenze che tentano di "uscire" dall'isola. Questo approccio cattura generalizzazioni cross-linguistiche sulle isole senza ricorrere a stipulazioni specifiche per ogni costruzione.

Implementazioni Computazionali e Parsing Algorithms

L'implementazione computazionale dei formalismi categoriali richiede algoritmi efficienti per il parsing e la generazione. Il problema fondamentale del parsing categoriale è: data una sequenza di categorie lessicali, determinare se esiste una derivazione valida che produce una categoria obiettivo (tipicamente S per frase). Questo problema è decidibile per tutti i sistemi categoriali standard, ma la complessità computazionale varia significativamente tra diversi formalismi.

Per il calcolo di Lambek puro (senza prodotto), il parsing è NP-completo, come dimostrato da Pentus. Questo significa che nel caso peggiore, il tempo richiesto per determinare la derivabilità può crescere esponenzialmente con la dimensione dell'input. Tuttavia, per frammenti ristretti e con grammatiche specifiche, algoritmi più efficienti sono possibili. In particolare, l'algoritmo di chart parsing può essere adattato al calcolo di Lambek, usando programmazione dinamica per evitare di ricalcolare le stesse sotto-derivazioni multiple volte.

L'algoritmo CKY (Cocke-Kasami-Younger), originariamente sviluppato per grammatiche context-free in forma normale di Chomsky, può essere generalizzato al parsing categoriale. L'idea è costruire una tabella (chart) che memorizza tutte le derivazioni possibili per ogni sottostringa della frase di input. La tabella è riempita in modo bottom-up, partendo da sottostringhe di lunghezza 1 (singole parole) e costruendo progressivamente derivazioni per sottostringhe più lunghe combinando derivazioni di sottostringhe più corte.

Per il calcolo di Lambek, l'algoritmo procede come segue. Per ogni posizione i e j nella frase (con $i < j$), e per ogni categoria A , determiniamo se la sottostringa dalle posizioni i a j può ricevere la categoria A . Questo è vero se: (1) la sottostringa consiste di una singola parola che ha categoria lessicale A , oppure (2) esiste una posizione intermedia k e categorie B e C tali che la sottostringa da i a k ha categoria B , la sottostringa da k a j ha categoria C , e B e C possono combinarsi via applicazione funzionale o composizione per produrre A .

La complessità di questo algoritmo è $O(n^3 \times |G|^3)$ dove n è la lunghezza della frase e $|G|$ è la dimensione della grammatica (numero di categorie). Il fattore n^3 deriva dal fatto che dobbiamo considerare tutte le coppie di posizioni i, j e tutte le possibili split points k . Il fattore $|G|^3$ deriva dal fatto che per ogni split dobbiamo considerare tutte le possibili triple di categorie A, B, C . Nella pratica, questo può essere ottimizzato usando tecniche di indexing e pruning.

Per le grammatiche categoriali combinatorie, algoritmi simili possono essere sviluppati, ma devono tenere conto delle regole aggiuntive come la composizione funzionale e il type-raising. Un approccio è usare un algoritmo di chart parsing generalizzato che mantiene non solo le categorie derivate ma anche le lambda-termini semantici associati. Questo permette di costruire simultaneamente sia la derivazione sintattica che l'interpretazione semantica.

Il parsing probabilistico per grammatiche categoriali usa modelli statistici per assegnare probabilità alle diverse derivazioni possibili, permettendo di selezionare la derivazione più probabile in caso di ambiguità. I modelli possono essere addestrati su corpora annotati (treebanks) usando tecniche di maximum likelihood estimation o algoritmi più sofisticati come l'expectation-maximization (EM) che possono apprendere da dati non completamente annotati.

Il CCGbank, un corpus annotato con categorie CCG derivato dal Penn Treebank, ha reso possibile lo sviluppo di parser statistici CCG su larga scala. Questi parser usano modelli log-lineari o reti neurali per stimare le probabilità delle categorie lessicali e delle regole combinatorie. Il parser di Clark e Curran, uno dei più influenti, raggiunge accuratèzze competitive con parser basati su altre grammatiche formali, tipicamente intorno all'85-90% di F-score su constituency.

I parser neurali per grammatiche categoriali hanno recentemente fatto progressi significativi sfruttando architetture di deep learning. Questi parser tipicamente usano embeddings contestuali da modelli pre-addestrati come BERT per rappresentare le parole, e poi usano reti neurali ricorrenti, convoluzionali o basate su attenzione per predire le categorie e le derivazioni. Un vantaggio di questi approcci è che possono catturare dipendenze a lungo raggio e interazioni complesse che sono difficili da modellare con features discrete tradizionali.

Il supertagging, introdotto da Bangalore e Joshi, è una tecnica che semplifica il parsing categoriale predicendo prima le categorie lessicali (supertags) per ogni parola, e poi usando queste assegnazioni per guidare il parsing. Un supertag è essenzialmente una categoria categoriale complessa che codifica non solo la categoria sintattica di una parola ma anche informazioni sulle sue dipendenze. Il supertagging riduce significativamente lo spazio di ricerca del parser, poiché invece di considerare tutte le possibili categorie per ogni parola, il parser considera solo quelle predette dal supertagger.

I moderni supertagger neurali raggiungono accuratèzze molto elevate, tipicamente sopra il 95% per l'inglese, rendendo il supertagging un preprocessing estremamente efficace per il parsing. Con supertags accurati, il parsing può spesso essere completato in tempo quasi lineare, poiché le scelte locali fortemente vincolano le possibilità globali.

Sviluppi Recenti e Prospettive Future

Gli ultimi decenni hanno visto una convergenza interessante tra diverse tradizioni di ricerca nella sintassi formale, con la grammatica categoriale che ha incorporato intuizioni da approcci basati su vincoli, da formalismi lessicalisti e da modelli computazionali. Questa

convergenza ha portato allo sviluppo di framework ibridi che cercano di combinare i punti di forza di diversi approcci teorici.

La semantica distribuzionale, basata sull'idea che il significato delle parole può essere rappresentato attraverso vettori in spazi ad alta dimensione derivati da statistiche di co-occorrenza in grandi corpora, ha iniziato a interagire produttivamente con la semantica formale categoriale. Modelli composizionali distribuzionali cercano di combinare vettori di significato in modi che rispettano la struttura sintattica, usando operazioni come prodotto tensoriale, moltiplicazione elemento per elemento, o reti neurali per comporre rappresentazioni vettoriali secondo derivazioni categoriali.

I modelli di linguaggio neurale su larga scala come BERT, GPT e loro successori hanno dimostrato capacità impressionanti di catturare regolarità sintattiche e semantiche senza grammatiche esplicite. Questo ha sollevato questioni profonde sul ruolo delle grammatiche formali nella modellazione del linguaggio: sono necessarie le strutture simboliche esplicite, o possono le reti neurali catturare implicitamente le stesse generalizzazioni? La ricerca attuale sta esplorando modi per integrare induzione di grammatiche con apprendimento neurale, per esempio inducendo grammatiche categoriali da rappresentazioni neurali o usando grammatiche per guidare l'architettura delle reti neurali.

La semantica vettoriale compositiva (compositional distributional semantics) sviluppata da Bob Coecke, Mehrnoosh Sadrzadeh, Stephen Clark e collaboratori, usa la teoria delle categorie per fornire un framework rigoroso per comporre vettori di significato. In questo approccio, le categorie sintattiche sono interpretate come oggetti in una categoria compatta chiusa, e le operazioni sintattiche come morfismi che compongono spazi vettoriali. Il prodotto tensoriale è usato per combinare vettori secondo la struttura sintattica, e il tipo di un verbo transitivo, per esempio, determina come i vettori del soggetto e dell'oggetto sono combinati per produrre il vettore della frase.

I modelli quantistici del linguaggio, ispirati dalla meccanica quantistica, usano spazi di Hilbert e operatori lineari per rappresentare significati e operazioni sintattiche. Questi modelli possono catturare fenomeni come l'ordine delle parole e interferenza semantica che sono difficili da modellare in spazi vettoriali classici. La categoria dei spazi di Hilbert finito-dimensionali è compatta chiusa, rendendola ideale per interpretare grammatiche categoriali con semantica vettoriale.

La semantica pragmatica e la teoria dei giochi hanno iniziato a essere integrate con grammatiche formali per modellare fenomeni di uso del linguaggio in contesto. La Rational Speech Act theory, per esempio, modella il significato come emergente da ragionamento ricorsivo tra parlante e ascoltatore su quale segnale linguistico scegliere per comunicare una particolare intenzione. Questo approccio può essere combinato con grammatiche categoriali per modellare come la struttura sintattica interagisce con inferenze pragmatiche.

Le applicazioni pratiche delle grammatiche categoriali si sono espanse oltre il parsing tradizionale. Nel question answering semantico, le grammatiche categoriali sono usate per parsare domande in linguaggio naturale in query logiche che possono essere eseguite su knowledge bases. Nella generazione di linguaggio naturale, le grammatiche forniscono templates strutturati che garantiscono grammaticalità e permettono controllo fine sulla

struttura dell'output. Nella traduzione automatica, le *synchronous grammars* permettono di mantenere corrispondenze strutturali tra lingue.

I *dialogue systems* e gli assistenti conversazionali stanno iniziando a incorporare grammatiche formali per migliorare la robustezza e l'interpretabilità. Mentre i modelli *end-to-end* neurali possono generare risposte fluenti, spesso mancano di capacità di ragionamento compositivo rigoroso. L'integrazione di parsing categoriale con modelli neurali può permettere sistemi che combinano la flessibilità dell'apprendimento statistico con le garanzie formali delle grammatiche simboliche.

La verifica formale di sistemi di elaborazione del linguaggio naturale è un'area emergente dove le grammatiche categoriali, con le loro fondazioni logiche solide, giocano un ruolo importante. Usando assistenti alla dimostrazione come Coq o Agda, è possibile specificare grammatiche formali e dimostrare proprietà dei parser (come correttezza e completezza) con garanzie matematiche. Questo è particolarmente importante per applicazioni *safety-critical* dove errori di interpretazione possono avere conseguenze serie.

La linguistica computazionale multilingue ha beneficiato dell'approccio categoriale grazie alla sua flessibilità nel modellare diverse tipologie linguistiche. Le grammatiche categoriali astratte permettono di separare la struttura logica universale dalla realizzazione superficiale specifica di una lingua, facilitando lo sviluppo di sistemi che funzionano su multiple lingue con core semantico condiviso ma sintassi variabile. Questo approccio è stato usato con successo in progetti come *GrammaticalFramework* (GF) per costruire grammatiche parallele per dozzine di lingue.

Le questioni di efficienza computazionale rimangono centrali. Nonostante i progressi algoritmici, il parsing con grammatiche molto espressive rimane computazionalmente costoso. La ricerca continua su algoritmi approssimativi, euristiche di pruning, e architetture parallele che possono sfruttare hardware moderno (GPU, TPU) per accelerare il parsing. I metodi di parsing neurali, che convertono il parsing in problemi di classificazione o *sequence-to-sequence* che possono essere risolti efficientemente con reti neurali, rappresentano una direzione promettente.

Connessioni con Altri Domini Scientifici

Le grammatiche categoriali e la teoria dei tipi hanno trovato applicazioni e connessioni con aree scientifiche apparentemente distanti dalla linguistica. La biologia computazionale, per esempio, ha usato grammatiche formali per modellare la struttura secondaria dell'RNA e le interazioni proteiche. Le *stochastic context-free grammars* sono usate per predire il folding dell'RNA, dove le regole grammaticali corrispondono a pattern di appaiamento di basi. Recentemente, approcci categoriali sono stati esplorati per modellare pathway biologici e reti di interazione molecolare.

In fisica teorica, la teoria delle categorie è diventata un linguaggio fondamentale per formulare teorie topologiche di campo quantistico e per studiare dualità in teoria delle stringhe. Le categorie monoidali simmetriche che appaiono nella fisica quantistica sono strutturalmente simili a quelle usate nella semantica categoriale del linguaggio, suggerendo

analogie profonde tra composizione fisica e composizione linguistica. Alcuni ricercatori hanno speculato che principi categoriali universali potrebbero sottendere entrambi i domini.

Nella teoria della computazione e dei linguaggi di programmazione, la corrispondenza di Curry-Howard e la semantica categoriale hanno avuto un impatto trasformativo. Linguaggi funzionali moderni come Haskell sono esplicitamente basati su teoria dei tipi e teoria delle categorie, con concetti come monadi, funtori e trasformazioni naturali che sono primitivi del linguaggio. La ricerca su type systems avanzati per linguaggi di programmazione e la ricerca su sistemi di tipi per linguaggio naturale si informano reciprocamente, con tecniche sviluppate in un dominio che trovano applicazioni nell'altro.

La teoria dell'informazione e la complessità algoritmica offrono prospettive quantitative sulla struttura linguistica. La complessità di Kolmogorov di una stringa, definita come la lunghezza del programma più corto che la genera, fornisce una misura della complessità intrinseca che può essere messa in relazione con la complessità delle derivazioni grammaticali. Grammatiche che producono derivazioni più compatte possono essere viste come fornendo compressioni migliori della struttura linguistica, con implicazioni per l'apprendibilità e l'elaborazione cognitiva.

Le scienze cognitive e le neuroscienze hanno iniziato a investigare se le strutture grammaticali formali hanno correlati neurali. Studi con fMRI e EEG cercano di identificare pattern di attivazione cerebrale associati con diverse operazioni sintattiche. Alcuni ricercatori hanno proposto che operazioni come merge nella grammatica generativa o applicazione funzionale nelle grammatiche categoriali potrebbero corrispondere a computazioni neurali specifiche, anche se queste connessioni rimangono speculative.

Questioni Filosofiche e Fondazionali

Le grammatiche categoriali sollevano questioni filosofiche profonde sulla natura del linguaggio e della cognizione. La questione della realtà psicologica delle strutture grammaticali formali è stata dibattuta estensivamente: le derivazioni categoriali rappresentano processi cognitivi reali durante la comprensione e produzione del linguaggio, o sono semplicemente strumenti formali per descrivere regolarità linguistiche? La distinzione tra competence (conoscenza linguistica astratta) e performance (uso effettivo del linguaggio) diventa problematica quando consideriamo modelli computazionali dettagliati.

Il principio di composizionalità stesso, centrale per le grammatiche categoriali, è stato oggetto di scrutinio filosofico. Mentre sembra intuitivamente plausibile che il significato di un'espressione complessa dipenda dai significati delle parti, ci sono fenomeni (idiomi, metafore, implicature) che sembrano resistere a analisi composizionali semplici. I difensori della composizionalità argomentano che questi fenomeni possono essere ricondotti a composizionalità a livelli appropriati di analisi, mentre i critici suggeriscono che la composizionalità è al meglio un'idealizzazione che approssima piuttosto che catturare esattamente la realtà linguistica.

La questione dell'innatismo e dell'apprendibilità è rilevante per le grammatiche formali. Se le grammatiche categoriali catturano correttamente la struttura del linguaggio, come vengono acquisite dai bambini? Il poverty of stimulus argument, che afferma che i bambini non

ricevono evidenza sufficiente dall'input per indurre la grammatica corretta, ha motivato ipotesi di conoscenza innata specifica del dominio. Tuttavia, progressi recenti in modelli computazionali di acquisizione del linguaggio, particolarmente usando deep learning, suggeriscono che più struttura può essere appresa dall'input di quanto precedentemente pensato.

La relazione tra sintassi e semantica è concepita diversamente in diverse tradizioni teoriche. Mentre le grammatiche categoriali tendono a vedere sintassi e semantica come proceeding in parallel, con una derivazione sintattica che determina univocamente un'interpretazione semantica, altri approcci permettono maggiore autonomia tra i due livelli. La questione è se la sintassi è in qualche senso più fondamentale, con la semantica che interpreta strutture sintattiche pre-esistenti, o se sintassi e semantica sono meglio viste come due facce della stessa struttura sottostante.

Il realismo versus strumentalismo riguardo alle entità grammaticali è un'altra questione filosofica. Le categorie sintattiche, i tipi semantici e le operazioni combinatorie sono entità reali che esistono nella mente/cervello dei parlanti, o sono costrutti teorici utili che organizziamo per predire e spiegare comportamento linguistico? Questa questione tocca dibattiti più ampi nella filosofia della scienza sulla natura delle teorie scientifiche e dello status ontologico delle entità teoriche.

Conclusioni e Sintesi

La sintassi categoriale e la teoria dei tipi nel linguaggio naturale rappresentano un programma di ricerca maturo e multifacetico che ha prodotto intuizioni profonde sulla struttura del linguaggio e sulle relazioni tra sintassi, semantica e computazione. Partendo dalle idee seminali di Ajdukiewicz e Bar-Hillel, passando attraverso le innovazioni logiche di Lambek e le estensioni lessicaliste di Steedman, fino agli sviluppi contemporanei che integrano machine learning, teoria delle categorie e semantica distribuzionale, questo paradigma ha dimostrato notevole vitalità e capacità di adattamento.

I punti di forza dell'approccio categoriale includono il suo rigore formale, la stretta integrazione di sintassi e semantica, la base logica solida che permette ragionamento formale e verifica, e la flessibilità nel modellare diverse tipologie linguistiche. La corrispondenza di Curry-Howard fornisce un ponte concettuale elegante tra logica, computazione e linguaggio, rivelando unità profonde tra domini apparentemente disparati. Le applicazioni pratiche in parsing, generazione, question answering e traduzione dimostrano che questi formalismi non sono solo eleganti matematicamente ma anche computazionalmente utili.

Le sfide rimangono. La complessità computazionale del parsing per grammatiche espressive continua a essere un ostacolo per applicazioni su larga scala. L'ambiguità proliferante e la necessità di disambiguazione statistica richiedono integrazione con modelli probabilistici che può complicare le fondazioni formali pulite. Fenomeni semantici complessi come idiomi, metafore e implicature continuano a sfidare analisi composizionali rigorose. La questione della realtà psicologica e del ruolo dell'apprendimento versus conoscenza innata rimane aperta.

Guardando al futuro, le direzioni più promettenti sembrano coinvolgere ibridazioni creative: tra approcci simbolici e neurali, tra grammatiche formali e modelli statistici, tra competenza idealizzata e performance situata. L'integrazione di grammatiche categoriali con embeddings distribuzionali e architetture neurali profonde offre la possibilità di sistemi che combinano composizionalità rigorosa con robustezza statistica. L'applicazione di assistenti alla dimostrazione e metodi formali all'elaborazione del linguaggio può portare a sistemi verificati con garanzie di correttezza. L'esplorazione di connessioni con domini come la fisica quantistica e la biologia può rivelare principi composizionali universali che trascendono domini specifici.

In definitiva, le grammatiche categoriali e la teoria dei tipi offrono non solo strumenti tecnici per l'analisi linguistica ma anche una filosofia del linguaggio: la visione che il significato emerge da combinazione sistematica secondo principi composizionali rigorosi, che la struttura sintattica e il contenuto semantico sono intimamente connessi, e che il linguaggio può essere compreso attraverso le lenti della logica, della computazione e della matematica. Questa visione continua a ispirare ricerca teorica e applicazioni pratiche, promettendo ulteriori intuizioni sulla natura del linguaggio e della cognizione umana.